VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Nástroj pro generování rostlin a stromů

Tree and plants modeling software

2015 David Kubát

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **David Kubát**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Nástroj pro generování rostlin a stromů
Tree and Plants Modeling Software

Zásady pro vypracování:

V současné době existuje velká řada modelovacích nástrojů pro tvorbu 3D scén a to jak komerčních tak nekomerčních. Součástí této práce je vytvoření nástroje, který bude umožňovat generovat rostliny, keře a stromy. Objekty se budou generovat na základě vstupních parametrů (gravitace, výška apod.) a bude možné je exportovat pro použití v dalších systémech.

1. Nastudujte problematiku generování rostlin a stromů založených na biologicky inspirovaných algoritmech.
2. Vytvořte nástroj umožňující vygenerovat rostliny, keře a stromy, s možnosti jejich výsledné editace. Využijte externích profesionálních nástrojů (Blender, 3D studio apod.).
3. Vyberte vhodný formát (FBX), ve kterém půjde výsledné vygenerované objekty uložit tak, aby je bylo možné použít v dalších nástrojích (např. UE4).
4. V případě generovaných objektů půjde navolit také vlastnosti jednotlivým částem, jako jsou například jednotlivé textury, koncové listy stromů, květy rostlin, plody stromů apod.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Martin Němec, Ph.D.**

Datum zadání: 01.09.2014
Datum odevzdání: 07.05.2015

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky OSTRAVA 1

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26 odst. 9 *Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava*.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 6. května 2015

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomohli, protože bez nich by tato práce nevznikla.

Abstrakt

Cílem této bakalářské práce bylo zmapovat nástroje pro generování stromů. Teoretická část popisuje některé známé nástroje. V praktické části je vybrána metoda Self-organizing tree models for image synthesis [18], která je optimalizována a implementována do objektu scény Unreal Engine.

Klíčová slova: stromy, generování stromů, Unreal Engine, Cienma 4D, fbx, bakalářská práce

Abstract

Puprose of my bachelor thesis was to map tree generating tools. The teoretical part describe some popular tools. The practical part is implementation of Self-organizing tree models for image synthesis [18], witch was optimized and implemented as scene actor for Unreal Engine.

Keywords: trees, tree generator, Unreal Engine, Cinema 4D, fbx, bachelor thesis

Seznam použitých zkratek a symbolů

Atraktor	–	Bod v prostředí, pomocí kterého se určuje směr růstu.
Překážka	–	Těleso zamezující růstu stromu v dané oblasti.
Pupen	–	Základní element stromu. Udržuje si svou pozici, hodnotu dopadajícího světla, směr ve kterém vyrostl, směr ve kterém bude pokračovat růst a indikátor, zda-li již pupen neumřel. Dále obsahuje odkaz pupen, ze kterého vznikl, následující pupen ve větvi a první pupen podvětve, pokud došlo k větvení.
Kořen	–	Jedná se o výchozí pupen stromu, tzn. odkaz předchozího pupenu je prázdný.
Koncový pupen	–	Pupen, na který nenavazuje žádný další.
C4D	–	MAXON CINEMA 4D, grafické studio.
UE4	–	Unreal Engine 4, herní engine.
Fbx	–	Formát od firmy Autodesk pro ukládání 3D scén.
Aktér	–	Jakýkoli objekt umístěný do scény v editoru.
Tag	–	String, který je přiřazen k objektu. Slouží ke snadné identifikaci objektu.
Blueprint	–	Základní objekt vizuálního skriptovacího jazyka vytvořeného pro Unreal Engine 4.
LOD	–	Level of detail. Jedná se o úrovně detailů modelu, např. chybějící prvky modelu nebo méně kroků při zaoblování.

Obsah

Seznam obrázků

1 Úvod

Cílem této práce je zmapovat současnou situaci v oblasti programů ke generování modelů stromů a rostlin.

Dnes se tyto modely používají v mnoha odvětvích, například ve filmovém průmyslu, vizualizacích městské i bytové architektury nebo počítačových hrách. Komplikovanost algoritmů a výsledných modelů určuje možnost jejich použití. Pokud je algoritmus pomalý a generuje komplikované a věrohodné modely, je vhodný pro fotorealistickou scénu, kde si můžeme dovolit vyšší výpočetní čas než několik milisekund. Realtimové použití naopak potřebuje stromy s co nejmenším počtem polygonů a je vhodné vygenerovat model s několika úrovněmi detailů, aby vzdálené stromy, které zabírají malou plochu na obrazovce, zbytečně nezatěžovaly vykreslování.

V praktické části jsem se inspiroval metodou Self-organizing tree models for image synthesis 3.2.2 a naprogramoval jsem podobnou metodu generování stromů. Vytvořil jsem plugin pro program Maxon CINEMA 4D R16 a objekt umístitelný do scény v Unreal 4 Editoru. Jako ukázku jsem vytvořil program v Unreal Engine 4.7 [11], který umožňuje si vyzkoušet objekt v akci a výsledný strom vyexportovat jako .fbx soubor.

2 Typy renderování a jejich požadavky

Velmi záleží na způsobu použití modelu, cíli práce a dostupných prostředcích. Pokud rendering není omezen výpočetním časem, bude se jednat o fotorealistický rendering. Více v kapitole 2.1. Ten dokáže renderovat stromy v kvalitě nerozeznatelné od reality. Pokud však chceme tvořit model pro hry nebo jinou aplikaci, která potřebuje aktualizovat obraz v reálném čase, je nutné dodržet požadavky realtime renderingu. Více v kapitole 2.2.

2.1 Fotorealistický rendering

Při tomto renderování jsou dostupné velké hardwarové prostředky a příliš nezáleží na výpočetním čase, proto má kvalita nejvyšší prioritu. Modely jsou velmi přesné, zahrnují detaily jako praskliny kůry, různé boule, malé uschlé větve, popřípadě plody, květy nebo kmeny porostlé mechem a lišejníkem. Listy často bývají samostatný model s texturou listu, někdy i komplexní model přesně ve tvaru listu skutečného stromu. Model je vhodné generovat celistvý a dodržet přesnou čtvercovou strukturu. Tím umožníme editaci kmene ve sculptovacích programech či programech pro malování přímo na geometrii modelu. Je možné použít komplikované materiály a tím zaručit velmi realistický vzhled.

Obrázek 1: Fotorealistický 3D render rostlin z programu XFrog [13]

2.2 Realtime rendering

I když je cílem mít strom co nejvíce podobný realitě, při realtime renderingu musíme zachovat framerate aplikace. Ten nesmí spadnout pod 30 snímků za sekundu a optimálně se snažíme udržet alespoň 60 snímků za sekundu, tzn. máme zhruba 16 milisekund na vykreslení všech objektů ve scéně. Proto jsou nutné optimalizace modelů a materiálů. V minulosti bylo nutné redukovat počet polygonů natolik, že stromy byly složeny pouze ze dvou ploch, které byly na sebe kolmé, potažené texturou. Dnes je hardware několika násobně výkonnější, ale stále ne dost, aby se nemusely dělat žádné ústupky. Ale právě díky tomuto nárůstu výkonu je možné mít stromy o statisících polygonů, které se pohybují a vypadají, že do nich fouká vítr.

Úroveň detailů na stromu je nutné měnit podle jeho vzdálenosti od kamery a času, jak dlouho bude strom na obrazovce. Pokud například tvoříme závodní hru a stromy jsou jako kulisa rychle projíždějícím vozům, detaily nejsou důležité, stromy jsou často postaveny pouze z několika polygonů, někdy nemají ani plnohodnotný kmen a celý vzhled je určen texturami s alfa kanálem. Také není nutné dělat mnoho různých modelů. Stačí stejný model přenásobit různým měřítkem a stromy různě natočit. Zde také není nutné stromy animovat, protože na scéně budou velmi krátký čas a uživatel se svým zrakem bude více soustředit na svůj vůz, trať a soupeře na ní.

Obrázek 2: Ukázka krajiny ze závodní hry Project C.A.R.S. [8]

Pokud však kamera bude mít možnost stromy pozorovat z malé vzdálenosti po delší dobu, tato zjednodušení jsou snadno pozorovatelná. A proto je lepší stromy vygenerovat v několika úrovních detailů, kdy při malé vzdálenosti je použit nejdetailnější model, který obsahuje i animace, a vzdálené stromy jsou pouze siluetou na obdélníku ze dvou polygonů bez animace tzv. billboardem. Přechod mezi jednotlivými úrovněmi detailů musí být nenápadný a strom musí mít stejnou siluetu v každé úrovni.

Další možnou optimalizací je nahradit některé větve jednoduchou texturou s listy místo složité geometrie. Válce kmene mohou mít malý počet segmentů a jemné chyby lze zamaskovat stíny a mlhou. Textury nepotřebují vysoké rozlišení a jemné detaily jsou simulovány normálovou mapou.

V některých hrách a simulacích je možno demolovat prostředí a v rámci toho lámat stromy nebo je celé vyvrátit i s kořeny. V těchto případech je velmi důležitá shoda topologie vzhledového modelu a kolizního modelu, aby padající strom nepropadal krajinou, a jiné dynamické objekty ve scéně reagovaly přímo na větve a kmen. Naopak pokud víme, že strom bude statický a kolize může nastat pouze u země, stačí kmen obalit kolizním válcem o jeho průměru.

Obrázek 3: Ukázka krajiny ze hry Witcher 3 [3]

3 Metody generování sromů

Existuje mnoho způsobů, jakými se dají získat vstupní data, jak je zpracovat a jak vytvořit výsledný model. Metody se liší úrovní realističnosti tvaru, výpočetního času a nároků na hardwarové prostředky. Výsledný vzhled však velmi závisí na nastavení materiálu, použitých textur a nasvětlení stromu.

3.1 Fraktálové systémy

Tyto systémy jsou založeny na bezkontextové gramatice. Jednotlivá slova určují tvar rostliny. Tyto systémy mají mnoho zástupců, malou hardwarovou náročnost a snadno se výsledky reprodukují. Jejich velkou nevýhodou je velmi obtížné zakomponování vlivu okolního prostředí na rostlinu, jako je například nedostatek místa, okolní rostliny nebo zastínění prostoru.

3.1.1 L-System

Základní algoritmus této skupiny, od kterého se odvíjí většina jiných fraktálových generátorů stromů. Jedná se o systém vyvinutý mad'arským biologem Aristidem Lindenmayerem v roce 1968 jako matematický formalizmus pro popisování růstu řas. Dnes se tímto systémem generují i jiné objekty, například ulice ve virtuálních městech. Výsledky této metody bez žádných úprav působí velmi "roboticky", jelikož je možno rozeznat často se opakující vzory.

Obrázek 4: Ukázka metody L-System [12]

3.1.2 Iterated function system

Jednoduchý fraktálový systém tvořící soběpodobné útvary. Výsledný tvar vzniká několikanásobným kopírováním stejného útvaru. Při každém kroku dojde ke zmenšení měřítka.

Obrázek 5: Ukázka IFS [5]

3.2 Bio inspirované algoritmy

Tyto algoritmy se snaží simulovat růst rostliny z vlivů prostředí. Výsledky těchto metod vypadají často velmi realisticky, avšak jejich náročnost na implementaci a hardware velmi rychle stoupají s počtem zakomponovaných vlivů. Dále díky velkému množství proměnných je často nutné uložit velké množství dat pro úspěšnou reprodukci výsledku. Tyto algoritmy nejsou často využívány právě z důvodu velké hardwarové náročnosti.

3.2.1 Modeling Trees with a Space Colonization Algorithm

Tato metoda bere v potaz pouze okolí stromu. Do určité oblasti je vygenerován náhodný počet atraktorů s náhodnou pozicí. Tyto atraktory určují, kudy strom poroste, proto je možné jejich odstraňováním dosáhnout obtékání objektů ve scéně, a je možné zaručit, že dvě větve nikdy nebudou procházet stejným bodem. Výsledky této metody při správném nastavení působí velmi realisticky, ale při nevhodném nastavení generuje velmi dlouhé větve působící nepřirozeně a stromy vypadají jako z hororového filmu. Originální dokument naleznete v seznamu literatury pod odkazem [14].

Obrázek 6: Ukázka metody Space colonization [9]

3.2.2 Self-organizing tree models for image synthesis

Tato metoda rozšiřuje metodu Space colonization 3.2.1 o stínovou mapu, určující o kolik strom poroste a zda-li nejsou větvě pro strom již zbytečné. Pokud na větev nedopadá dostatek světla, je větev označená jako mrtvá, dále neroste a není ve výsledném modelu použita. Dále je změněna práce s atraktory. Odstraňují se atraktory v určité vzdálenosti od jednotlivých pupenů a vliv atraktorů je omezen úhlem. Tento úhel zaručuje, že větev nebude drasticky měnit směr růstu. Tato metoda produkuje mnohem realističtější model stromu než Space colonization 3.2.1, avšak díky stínové mapě má velké nároky na paměť. Originální dokument naleznete v seznamu literatury pod odkazem [18].

Obrázek 7: Ukázka metody Self-organizing tree models for image synthesis [18]

3.3 Rekonstrukční algoritmy

Tento typ algoritmů se používá, pokud chceme přesné modely. Hodí se například pro provedení rekonstrukce existující krajiny do 3D modelu. Vstupem jsou fotografie skutečných stromů nebo data ze 3D skeneru. Metody v této kategorii jsou si velmi podobné a jejich výsledky jsou téměř identické s reálnými stromy, které sloužily jako zdroj vstupních dat.

3.3.1 Image-based Tree Modeling

Tato metoda využívá jako vstup deset až dvacet fotografií vzorového stromu a z nich sestaví pole bodů určující jeho obrys. Pomocí tohoto pole bodů rekonstruuje viditelné větve a ty zakryté dopočítá. Listy se také rekonstruují z těchto fotografií, ale nejsou generovány list po listu, nýbrž jako shluk listů, kterému se přiřadí část fotografie jako textura. Originální dokument naleznete v seznamu literatury pod odkazem [17].

Obrázek 8: Ukázka metody Image-based Tree Modeling [17]

4 Existující programy na generování stromů

Výběr existujících nástrojů je rozmanitý, od malých, schopných generovat jeden typ rostlin, až po nástroje generující celé lesy. Zde najdete seznam některých z nich a popis těch, se kterými jsem pracoval. Velký list těchto nástrojů je také na stránkách VTPSoftware [7].

4.1 Seznam nástrojů

XFrog	http://xfrog.com
SpeedTree	http://www.speedtree.com
Maya Paint Effects	http://www.autodesk.com/products/maya/overview
Arbaro	http://arbaro.sourceforge.net
Lenné3D	http://lenne3d.com
Intel Smoke Library	https://software.intel.com/en-us/articles/smoke-game-technology-demo
ngPlant	http://ngplant.sourceforge.net
Silvador	http://marketplace.bisimulations.com/silvador-pro/
Virtual Gardening and Virtual Bonsai	http://www.jfp.co.jp/garden_e/
Ivy Generator	http://graphics.uni-konstanz.de/ luft/ivy_generator/
OpenAlea	http://openalea.gforge.inria.fr/dokuwiki/doku.php
Treemagik	http://www.thegamecreators.com/?m=view_product&id=2087
Plant-Life	http://www.plantlife.org.uk
Plant Studio	http://www.kurtz-fernhout.com/PlantStudio/
Forester Arboretum	http://www.dartnall.f9.co.uk

4.2 Popis existujících nástrojů

Na komerční scéně existují dva velké nástroje. První z nich je XFrog. Ten je určen ke generování fotorealistických rostlin, stromů a trávy. Dokáže velmi dobře modely zakomponovat do prostředí pomocí vyhýbání se překážkám nebo plazení se po objektech scény. Rosliny lze rozpohybovat ve větru a animovat jejich růst. Zvládá komplexní scény s mnoha rostlinami a je skvělý pro vizualizaci architektonických projektů.

Nástroj je k dispozici jako plugin pro MAXON CINEMA 4D a Autodesk Maya nebo jako samostatný program za jednotnou cenu $189.50 na oficiálních stránkách XFrog [13]. Dále můžete na těchto stránkách zakoupit již hotové modely rostlin v mnoha formátech nebo textury či materiály.

Obrázek 9: Ukázka výsledků programu XFrog [13]

Druhým nástrojem je SpeedTree, který lze použít ke generování mnoha typů rostlin a stromů. Výstup je vhodný pro realtimové programy. Nástroj staví stromy pomocí různých generátorů, ze kterých se skládá výsledná rostlina. To umožňuje vysokou kontrolu nad výsledným vzhledem stromu. Jednotlivé generátory používají rozdílné algoritmy a vstupní parametry. Dále nástroj umožňuje na rostlinách simulovat pohyb ve větru, a tím přidat dojem přirozenosti, nebo simulovat jejich růst. Dále velkou výhodou pro realtimové aplikace je automatická generace LOD, díky které je možné mít na scéně mnoho stromů najednou.

Na internetové stránce https://store.speedtree.com je možné zakoupit modely tvořené tímto nástrojem nebo samostatný nástroj v několika verzích a edicích. Verze se pro filmy a animace prodává v následujících variantách: Edice Architect za $499, která umožňuje pouze generování geometrie stromu. Edice Studio v ceně $895.00, která navíc nabízí efekt pohybu ve větru a možnost obrůstání povrchu objektů. A nejdražší edice CINEMA za $4995.00, která obsahuje vlastnosti všech předešlý edicí a navíc knihovnu materiálů a textur, možnost animace růstu a změny ročního období. Pokud by uživatel

chtěl použít tento nástroj k tvoření obsahu do her nebo realtimových aplikací, jsou k dipozici pluginy pro populární herní enginy UE3, UE4 a Unity 5, kde se nástroj integruje přímo do editoru. Tato verze stojí $19 měsíčně a nelze ji koupit jako trvalou licenci.

Obrázek 10: Ukázka výsledků programu SpeedTree [10]

Za zmínku rozhodně také stojí nástroj Maya Paint Effects, který je zabudován do programu Autodesk Maya a slouží k přidávání objektů a efektů do scény. Obsahuje knihovnu se spoustou květin, trav, stromů, mraků a jiných speciálních vizuálních efektů. Rostliny je možné animovat a u některých simulovat růst. Nástroj však poskytuje velmi malou kontrolu nad výsledným vzhledem rostliny a nelze ji použít v jiných aplikacích. Protože nelze výsledky exportovat, je použití omezeno na scény a obrázky renderované pomocí Maya Software renderer.

Pokud chcete tento nástroj používat, musíte zakoupit program Autodesk Maya za cenu $3675.00 nebo si zaplatit měsíční předplatné $185.00.

Obrázek 11: Ukázka výsledků programu Maya Paint Effects [4]

Z open source programů bych doporučil nástroj Arbaro psaný v jazyku Java. Samotné generování je velmi podobné nástroji SpeedTree, není však k dispozici tak velká knihovna předdefinovaných materiálu a textur. Není možné stromy nijak animovat a chybí real-time náhled na výsledek. Nabízí export do .obj, a tím umožňuje použití v jiných programech, nástrojích a realtime aplikacích. Je vhodný, pokud tvoříte něco jednoduchého a nechcete investovat stovky nebo tisíce dolarů.

Program lze stáhnout včetně zdrojový kódů ze stránky SourceForge [1].

Obrázek 12: Ukázka výsledků programu Arbaro

5 Praktická část

Pro generování svých stromů jsem se rozhodl využít metody Self-organizing tree models for image synthesis 3.2.2. Článek o této metodě obsahoval teoretický popis jejího algoritmu. Článek působil velmi profesionálně a zajímavě. Tato metoda má dva základní kameny. Prvním z nich je simulace dopadajícího světla na větve stromu. K této simulaci se využívá voxelová mapa, do které se zapisuje zastíněný prostor. Druhou důležitou složkou jsou body v prostoru zvané atraktory. Ty jsou náhodně rozmístěny a určují směr růstu stromu. Pomocí vytvoření oblastí bez atraktorů je možné zaručit obtékání objektů ve scéně. Přesným definováním oblasti, kde se atraktory generují, můžeme vynutit tvar stromu a koruny. Byl jsem nucen provést několik změn oproti původní metodě. První změnou bylo nahrazení voxelové mapy pro stín dvěma parametry, které vytvoří stínová tělesa ve tvaru kuželu pod každým pupenem a výpočtem polohy pupenů proti stínovým tělesům. To sice mírně zvýšilo výpočetní čas zhruba v řádu desítek milisekund, ale snížilo paměťové nároky, které při velkých stromech nebo několika stromech činily až jednotky GB. Více v kapitole 5.3. Druhou změnou byl přepočet zdrojů v kořeni před šířením do větví. Tato změna je podrobně popsána v kapitole získání délky růstu (5.3). Třetí změnou je způsob větvení. Protože původnímu dokumentu chyběla jasně definovaná podmínka, rozhodl jsem se napodobit proleptické větvení. Větvení je podrobně popsáno v kapitole Růst větví a větvení (5.5). Celý algoritmus včetně úprav je popsán v následujících kapitolách a jsou připojeny odkazy k využité dokumentaci. Následuje několik ukázek toho, jak hodnoty vstupních parametrů do výpočtu ovlivňují vzhled výsledného objektu.

5.1 Generování atraktorů a prostředí

Jelikož je prostředí teoreticky nekonečné, musí být atraktory generovány dynamicky. Prostředí je rozděleno na krychle o konstantní velikosti uspořádané do pravidelné mřížky. Do těchto krychlí je vygenerován náhodný počet atraktorů s náhodnou pozicí uvnitř krychle. Meze počtu atraktorů jsou nastaveny uživatelskými parametry Atr_{MIN} (5.9) a Atr_{MAX} (5.9). Krychle se generují v okamžiku, kdy se algoritmus libovolným způsobem snaží pracovat s atraktory v prostoru, který je mimo dosud existující krychle. Tyto krychle jsou uloženy do pole a třídění podle polohy. To umožňuje vyhledávání pomocí půlení pole a sníží potřebný čas na nalezení krychle, se kterou chceme pracovat. Atraktory, které jsou algoritmem označeny za odstraněné jsou rovnou odstraňovány z paměti a tím je možné prostředí "nafukovat"do velkých rozměrů. Achillovou patou tohoto postupu je však samotné třídění pole, které při velkém prostředí trvá i několik stovek ms, a plánuji ho v budoucnu nahradit BSP stromem.

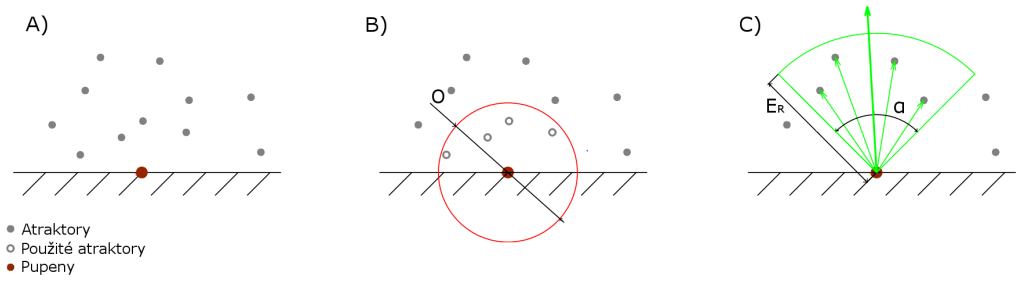
5.2 Získání směru růstu

Prvním krokem je odstranění všech atraktorů v prostoru okolo již existujících pupenů. Tento prostor je určen koulí se středem v pupenu a o rádiusu, určeném uživatelským parametrem O (5.9) (Obrázek 13 B). Tím se zamezí větvím procházet přes stejné místo.

Další krok sesbírá všechny atraktory nacházející se uvnitř koule se středem v pupenu a o rádiusu zadaném parametrem E_R 5.9. Pro tyto atraktory se spočítá vektor

$$\vec{V_a} = A - P_{pos} \tag{1}$$

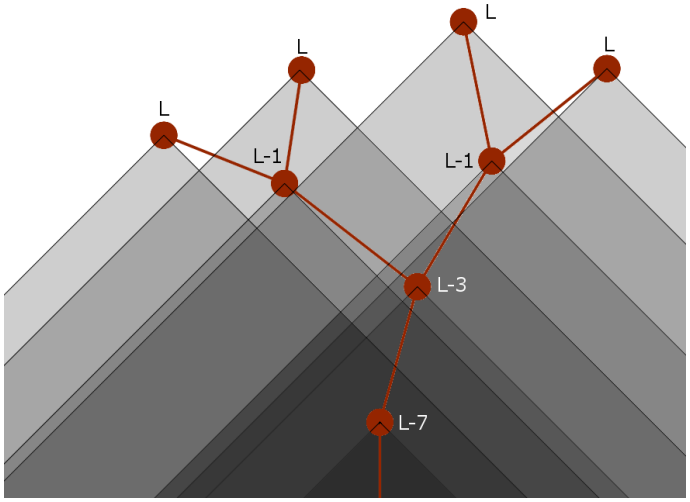
kde A je pozice atraktoru a P_{pos} pozice pupenu. Vektory $\vec{V_a}$, které mají úhel menší než parametr E_α 5.9 a neprotínají žádné kolizní těleso, jsou normalizovány, sečteny a výsledek znovu normalizován. Výsledek určí směr růstu rostliny. Pokud není nalezen žádný atraktor, splňující všechny tyto podmínky, je do směru uložen nulový vektor. (Obrázek 13 C)



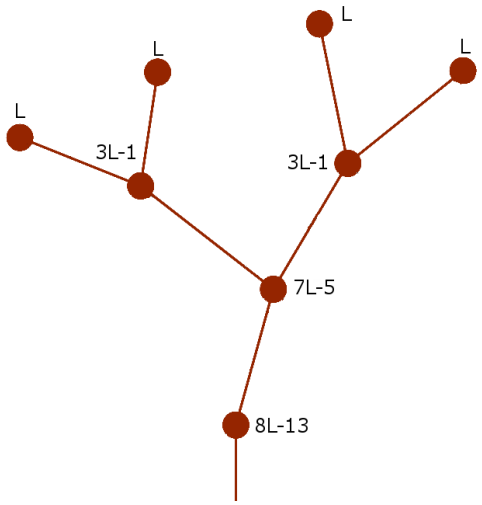
Obrázek 13: Způsob určení směru růstu

5.3 Získání délky růstu

Délka růstu je určena hodnotou zdrojů v pupenu, který má růst. Získání této hodnoty má tři fáze. První z nich je výpočet hodnoty dopadajícího světla na jednotlivé pupeny. Ta se získá jako rozdíl mezi parametrem L 5.9 a počtem stínových těles, která jsou tvořena pod každým pupenem ve tvaru kuželu, ve směru určeném parametrem S_{DIR} 5.9 a úhlem S_α 5.9. Pokud je výsledná hodnota menší než 0, jsou zdroje v pupenu nastaveny na 0.



Obrázek 14: Výpočet zdrojů v prvním kroku



Obrázek 15: Sečtení zdrojů do kořene

Poté se sčítají hodnoty světla od koncových pupenů směrem ke kořenu stromu. Hodnota světla pupenu v tomto kroku je rovna součtu hodnoty světla v pupenu z předešlého kroku a všech hodnot podřízených pupenů, které nejsou mrtvé. Hodnota, která vyjde v kořenech stromů se přepočítá pomocí vzorce:

$$P_{zk} = \frac{P_z N_i}{W + N_i} \tag{2}$$

kde P_{zk} je výsledné číslo zdrojů v kořeni, P_z jsou zdroje v kořeni z předcházejícího kroku, N_i je počet iterací simulace, W je počet živých větví stromu.

Ve třetím kroku se rozdělují zdroje zpět do větví. Postupuje se od kořene ke koncům větví pomocí vzorců

$$V_m = V \frac{\lambda Q_m}{\lambda Q_m + (1-\lambda) Q_l} \tag{3}$$

[18]

$$V_l = V \frac{(1-\lambda) Q_l}{\lambda Q_m + (1-\lambda) Q_l} \tag{4}$$

[18]

kde V_m hodnota zdrojů, které půjdou do primární větve, V_l hodnota zdrojů, které půjdou do vedlejší větve, V je hodnota zdrojů v pupenu, Q_m hodnota zdrojů, které jdou z primární větve, Q_l hodnota zdrojů, které jdou z vedlejší větve a λ je uživatelský parametr 5.9.

5.4 Odumírání větví

Při tomto kroku se určí, zda-li pupen má zdroje, aby pokračoval v růstu. Pokud jsou zdroje nulové, je u pupenu a všech jeho potomků nastaven parametr úmrtí na pravdivý.

5.5 Růst větví a větvení

Aby pupen mohl růst, musí se splnit následující podmínky: směr růstu musí být nenulový vektor, pupen nesmí být mrtvý, nesmí existovat hlavní nebo vedlejší potomek a pupen nebyl vytvořen při této iteraci. Nový pupen je vytvořen na pozici určené vzorcem:

$$P_{np} = P_p + \vec{P_s} P_z E - \vec{g} \tag{5}$$

kde P_{np} je výsledná pozice nového pupenu, P_p je pozice pupenu, pro který se počítá růst, $\vec{P_s}$ je směr růstu, vypočtený v kroku Získání směru růstu (5.2), P_z je hodnota zdrojů, spočítaná krokem Získání délky růstu (5.3), E je síla prostředí, zadána uživatelským parametrem E (5.9), $\vec{g}$ je gravitační vektor, zadaný uživatelským parametrem $\vec{g}$ (5.9). Směr, kudy

nový pupen vyrostl, a směr, kudy poroste, se nastaví na vektor $\vec{P_s}P_zE - \vec{g}$. Do původního pupene se přidá odkaz na nový pupen. Pokud je dosud odkaz na primární pupen prázdný, tak se uloží jako primární pupen. Pokud je odkaz na primární pupen obsazen, tak se ukládá jako sekundární pupen.

5.6 Výpočet průměru větví

Průměr větve v pupenu se počítá od koncových pupenů po kořen a nehledí se na to, zda je pupen mrtvý. Pokud pupen nemá žádného potomka, je tloušťka převzata z uživatelského parametru D (5.9). Jestliže pupen má pravě jednoho potomka, tak se jejich tloušťky rovnají. Poslední situace vzniká, pokud má pupen dva potomky, pak se tloušťka spočítá pomocí následujícího vzorce:

$$P_d = \sqrt[c]{P_{d1}^c + P_{d2}^c} \tag{6}$$

[18]

kde P_d je výsledná tloušťka větve, P_{d1} je tloušťka hlavního potomka, P_{d2} je tloušťka vedlejšího potomka a c je zadán uživatelským parametrem C (5.9).

5.7 Sestavení modelu

Model je generován od kořenů stromů po konce větví. Pro každý pupen, který není mrtvý, se vytvoří kruh vertexů se středem v pozici pupenu a v rovině zadané směrem, kudy pupen vyrostl, a pozicí pupenu. Tento kruh má průměr odpovídající tloušťce větve v pupenu. Dojde-li k větvení, je nutné vytvořit napojovací kruh vertexů. Napojovací kruh má pozici v místě dělení, ale normálový vektor roviny odpovídá směru růstu sekundární větve a průměr koncové tloušťce nové větve. Poté se všechny tyto kruhy spojí do válcovitých útvarů, tvořících kmen a větve. V místě koncových pupenu se vytvoří uzávěry ve tvaru kuželu.

5.8 Generování listů

Listy se generují ve všech pupenech, které nejsou mrtvé a mají tloušťku menší než uživatelský parametr 5.9 D_{MAX} . Pokud se jedná o koncový pupen, je list umístěn ve směru růstu. Pokud o pupen uprostřed stromu, je vybrán libovolný vektor kolmý na směr růstu v pupenu.

5.9 Uživatelem zadané parametry, jejich popis a optimální hodnoty

Zkratka	Jméno parametru	Popis	Optimální hodnoty
N	Počet iterací	Ovládá komplikovanost výsledného stromu.	10-18
L	Síla světla	Parametr určující rychlost růstu a hustotu stromu.	5 až 25
$\vec{g}$	Gravitace	Vektor simulující váhu větví.	{0, 0, 1} až {0, 0, -1}
E	Síla prostředí	Parametr určující rychlost růstu, slouží ke kompenzaci parametru "Síla světla".	1 až 10
S_{DIR}	Směr stínu	Vektor určující směr stínových kuželů tvořených pupeny.	{0, -1, 0}
S_α	Úhel stínového kuželu	Určuje úhel ve špičce stínového kuželu pod pupenem.	$60\,^\circ$ až $90\,^\circ$
O	Rádius prostoru obsazeného pupenem	Určuje rádius oblasti okolo pupenů, ve kterém jsou odstraněny atraktory.	O > LE
E_R	Rádius prostoru ovlivňující pupen	Určuje rádius oblasti okolo pupenů, ve kterém atraktory ovlivňují směr růstu.	O + 10 až O + 25
E_α	Úhel kuželu prostoru, který ovlivňuje pupen	Úhel ve špičce kuželu, ve kterém atraktory ovlivňují směr růstu.	$60\,^\circ$ až $90\,^\circ$
λ	λ	Určuje poměr zdrojů mezi hlavní a vedlejší větví. Hodnota je v rozmezí 0-1, přičemž 0 přiřadí všechny zdroje vedlejší větvi a 1 do hlavní větve.	0.5
C	Poměr tlouštěk při větvení	Hodnota určuje jak rychle roste tloušťka stromu a měla by se pohybovat mezi 2 a 3.	2.5
D	Tloušťka koncové větve	Tento parametr určuje tloušťku koncové větve a tím ovlivňuje tloušťku celého stromu.	0.5 až 1
D_{MAX}	Maximální tloušťka větve, na které rostou listy	Na větvích tlustších než tento parametr nerostou listy.	2R
Atr_{MIN}	Minimální počet atraktorů	Určuje minimální počet atraktorů v krychli prostředí.	2
Seg	Segmentace stromu	Určuje počet čtyřúhelníků, ze kterých jsou sestaveny válce větví.	5 - 12
Atr_{MAX}	Maximální počet atraktorů	Určuje maximální počet atraktorů v krychli prostředí.	64

5.10 Plugin pro MAXON CINEMA 4D R16

Maxon CINEMA 4D R16 je komplexní 3D modelovací a animační studio od německé firmy MAXON. S programem je dodáván také velmi pokročilý renderer CINEMA 4D Advanced Renderer, sculptovací nástroj a BodyPaint3D, který umožňuje malovat textury přímo na model.

Obrázek 16: Logo MAXON CINEMA 4D [6]

5.10.1 CINEMA 4D API [16]

Pro vytvoření pluginu je možné použit nativní C++ API, Python API, interní skriptovací jazyk COFFEE nebo grafický jazyk Xpresso. Pro svůj plugin jsem zvolil jazyk C++ z důvodu rychlosti a nejnižších nároků na paměť z nabízených jazyků.

C++ API má přístup k většině funkcí aplikace a je k dispozici plnohodnotný debuger včetně symbolových souborů pro všechny objekty a funkce aplikace. Velkou nevýhodou oproti jiným nabízeným jazykům je absence virtuálního stroje a izolace od hlavního procesu, to sice dovoluje vysokou rychlost, ale pokud je v kódu chyba, a dojde k vyhození výjimky, spadne celá aplikace, vypíše se stacktrace do souboru s logem a všechny neuložené změny jsou nenávratně ztraceny. U interpretovaných jazyků dojde pouze k zastavení běhu kódu pluginu a vypsání chybové hlášky do konzole a logu.

Jednotlivé pluginy mohou být několika typů. Tyto typy se liší svými dostupnými funkcemi, způsobem registrace a třídou, po které musí dědit. Pro bakalářskou práci jsem využil pluginu typu "Object Plugin", který umožňuje práci s objekty ve scéně. Tento typ může mít několik variant podle zadané masky při vytváření. Mnou použitá varianta se jmenuje OBJECT_GENERATOR. Ta umožňuje vytvářet vlastní geometrii. Plugin tohoto typu musí dědit po třídě ObjectData a implementovat metodu `virtual BaseObject * GetVirtualObjects(BaseObject *op, HierarchyHelp *hh)` v té se odehrává logika generování libovolné geometrie. Dále jsem implementoval metodu `virtual Bool Message(GeListNode *node, Int32 type, void *data)`, ta je volána, pokud dojde ke změnám ve scéně nebo je změněna hodnota v nastavení objektu. Je také volána, pokud dojde ke stlačení tlačítka v GUI vytvořené pluginem.

5.10.2 GUI

GUI je pro všechny jazyky a typy pluginů tvořeno pomocí .res souborů, které využívají speciální jazyk podobný C++. K dispozici je mnoho ovládacích prvků a pokud nenajdete, co potřebujete, je možné si napsat svůj vlastní ovládací prvek.

Jednotlivé prvky jsou označeny identifikátorem (číslem). Pomocí těchto identifikátorů se u prvku nastavují a načítají data a přiřazují stringy s popisem prvku. Tyto identifikátory je vhodné umístit do samostatného .h souboru, který je možné na includovat v hlavním kódu pluginu. API obsahuje i systém pro lokalizaci stringů, ve kterém se texty vyhledávají opět pomocí stejných identifikátorů.

5.10.3 Pozice kořenů

Kořeny jsou definovány pomocí libovolného objektu poskládaného z bodů (polygonový objekt, křivky atd.) a počáteční směr růstu určují normály v těchto bodech.

5.10.4 Geometrie modelu

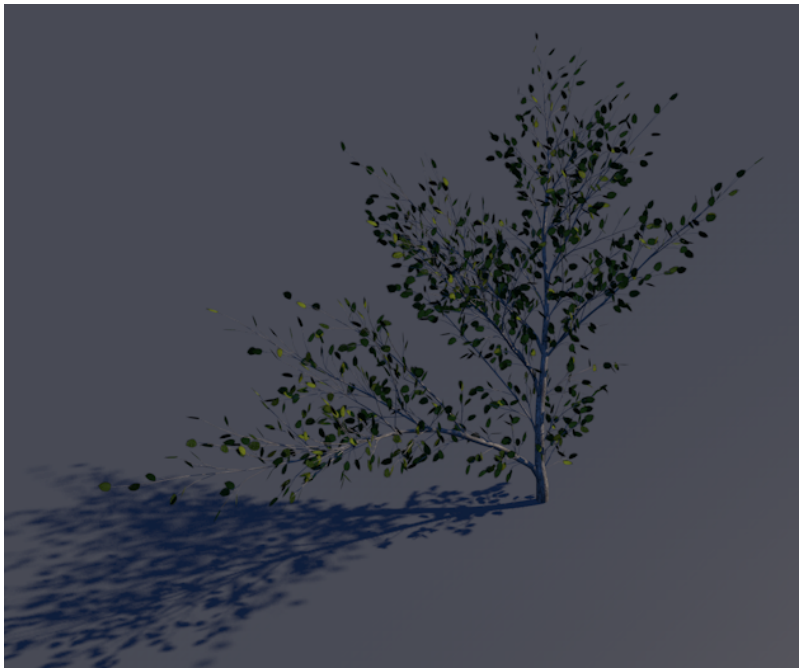
Model je vytvořen jako standardní polygonový objekt (třída PolygonObject) s Point Selection tagem pro listy se jménem Leafs, Phong tagem pro výpočet normál a UVW tagem pro UV koordináty. Válce větví jsou sestaveny ze čtyřúhelníků a uzávěry z trojúhelníků. Segmentace válců větví je určena uživatelským parametrem (5.9). UV koordináty jsou generovány přesně na velikost větve a jednotlivé větve se překrývají. Toto sice umožňuje snadné dlaždicování textury, ale pro kreslení na model je vhodné vytvořit druhou sadu UV koordinátů, kde se nebudou jednotlivé polygony překrývat.

5.10.5 Listy

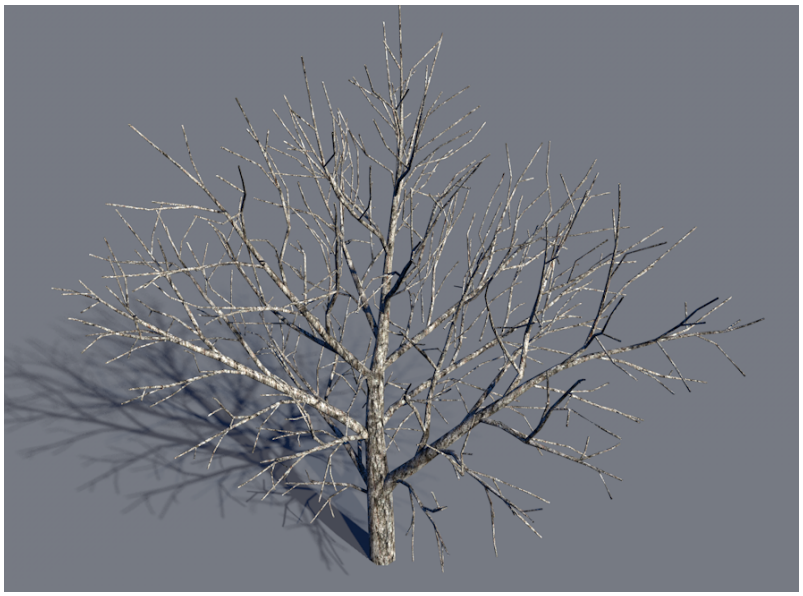
Geometrie listů je určená jedním nebo více objekty ve scéně. K vytvoření samotných listů využívám MoGraph Cloner Object, který je součástí CINEMA 4D ve verzích Broadcast a Studio. Ten slouží k naklonování geometrie listů na jejich místo. Pozice listů z modelu stromů se získá z tagu výběru bodů zvaného "Leafs".

5.10.6 Export

Protože je výsledný strom uložen jako standardní polygonový objekt, není žádný problém využít interního exportovacího nástroje do velkého výběru formátů (OBJ, FBX, Collada, atd.).



Obrázek 17: Ukázka výsledku pluginu pro Maxon CINEMA 4D R16



Obrázek 18: Ukázka výsledku pluginu pro Maxon CINEMA 4D R16

5.11 Objekt pro Unreal Editor 4

Unreal Editor 4 je program od společnosti Epic Games a slouží k vývoji softwaru založeném na Unreal Engine 4. Ten je jedním z třech největších a nejvyspělejších herních enginů na trhu. Kód bakalářské práce byl psán a testován na verzi 4.7.6.

Obrázek 19: Logo Unreal Engine 4

5.11.1 API [15]

Ke tvoření kódu pro Unreal Engine 4 je možno využít dva jazyky. Prvním z nich je neprogramátorsky přístupný grafický skriptovací jazyk nazvaný Blueprints Visual Scripting. Druhým jazykem je C++, který nabízí přístup ke všem veřejným funkcím enginu. K dispozici jsou také všechny zdrojové kódy enginu pro platformy Windows, Android a iOS. Zdrojové kódy pro konzole nejsou veřejně dostupné a dají se získat pouze s povolením firem vydávající danou konzoli (u PlayStation 4 se jedná o Sony a u Xbox One o Microsoft). Samotný engine je však stále ve fázi vývoje a aktualizace jsou velmi časté, proto pokud hledáte dokumentaci je důležité si ověřit, zda odpovídá vaší nainstalované verzi.

Unreal Engine 4 si přepisuje alokaci paměti svými algoritmy. Má vlastní dynamická pole, knihovnu pro sdílené pointry, a pro některé typy objektů dokonce garbage collection. Fyzickou paměť alokuje po blocích a neuvolňuje do vypnutí programu, drží si ji, i když jsou objekty smazány, a opět ji využívá při vytváření nových objektů. Tím si zaručí, že mu paměť během práce nevezme systém, nemusí svůj alokovaný blok paměti přesouvat a bude mít objekty vždy zarovnané k adrese dělitelné čtyřmi.

5.11.2 GUI

Gui je tvořeno pomocí UMG frameworku, který nabízí základní ovládací prvky, ale pokročilejší prvky je nutné si doprogramovat. Tyto prvky se dají vytvářet pouze jako Blueprint a nejsou dostupné pro jazyk C++. Gui se automaticky škáluje s rozlišením a dpi obrazovky.

5.11.3 Pozice kořenů

K definici kořenů slouží mnou vytvořený objekt zvaný TreeRootActor. Ten je vidět pouze v editoru a jeho osa +Z určuje směr růstu.

5.11.4 Definice kolizních těles a těles omezujících růst

Jako kolizní objekty se používají všichni aktéři, kteří dědí ze třídy UStaticMeshActor a mají tag "Obstacle". Jako tělesa omezujících růst se používají všichni aktéři, kteří dědí ze třídy UStaticMeshActor a mají tag "Bounding volume".

Kolizní objekty slouží k určení, kudy strom neporoste. Avšak do tohoto výpočtu není zahrnuta geometrie listů, proto jsem přidal uživatelský parametr M 5.11.10, o který se zvětší rozměry každého kolizního objektů, a tím se vytvoří bezpečná zóna.

Tělesa omezující růst určují, kde budou vygenerovány atraktory. Pokud žádné takovéto objekty nebyly do scény umístěny, jsou atraktory generovány bez omezení.

5.11.5 Listy

Pro každý bod, kde má růst list, se vytváří UStaticMeshComponent s náhodně vybraným meshem z pole LeafMeshes. Na UStaticMeshComponent se aplikují trasformace s náhodným pootočením a měřítkem ve všech třech osách. Limity pootočení jsou určeny uživatelskými parametry A_X 5.11.10 pro osu X, A_Y 5.11.10 pro osu Y a A_Z 5.11.10 pro osu Z. Limity pro měřítko jsou určeny uživatelským parametrem S 5.11.10 pro všechny tři osy.

5.11.6 Použití naprogramované komponenty

Komponentu ATreeGeneratorActor je vhodné použít jako předka pro nový blueprint, ve kterém si uživatel nastaví parametry. Poté tento blueprint umístit do scény nebo spawnout pomocí kódu. Pokud uživatel chce přegenerovat celý strom musí zavolat metodu **void** GenerateTree(). Ta spustí asynchronní operaci generování kostry stromu a přegeneruje model. Pomocí metody bool IsGenerating() lze zjistit, zda asynchronní operace stále běží nebo je již dokončena. Pokud uživateli stačí přegenerovat model, např. z důvodu změny průměru větví nebo změny listů, stačí zavolat metodu **void** GenerateModel().

5.11.7 Geometrie modelu

V tomto případě válce složené z čtyřúhelníků jsou rozsegmentovány na trojúhelníky, protože realtime aplikace jsou schopny vykreslovat pouze geometrii složenou z trojúhelníků. Dále bylo nutné ručně vygenerovat normály, tangentu a bitangentu. Normála je určena středem kruhu vertexů a polohou vertexu, ze kterého vychází. Tangenta je vytvořena ve směru růstu větve. Bitangenta je vytvořena vektorovým součinem normály a tangenty.

5.11.8 Ukázkový program

Program využívá komponenty ATreeGeneratorActor a pomocí UMG UI Frameworku zprostředkovává ovládání parametrů. Zde je možné si vyzkoušet vliv parametrů na vzhled stromů a dobu generování.

Jako modely listů jsem využil vzorky z knihovny SpeedTree dostupné zdarma. Nebyl jsem však schopen kompletně reprodukovat způsob animace pohybu větví a listů,

proto jsem naprogramoval vlastní shader, který animuje zeleň pomocí funkce integrované do Unreal Engine 4, sloužící k animaci trávy.

5.11.9 Export do FBX

K vytváření a práci s .fbx soubory jsem použil oficiální knihovnu Autodesk FBX SDK 2015.1 VS2013 [2]. Ta umožňuje vytvářet kompletní scénu s mnoha typy objektů. Ve své práci vytvářím jeden mesh pro každý strom. Tomuto meshi přiřadím geometrii odpovídající stromu a UV souřadnice pro difuzní složku. Listy jsou reprezentovány prázdnými objekty s transformací. Bohužel díky vnitřní struktuře prvků scény nejsem schopen vyexportovat modely listů a materiály s texturami. Textury jsou při kompilaci převedeny do speciálního formátu .uasset a zabaleny do archivu .pak, který je kontejner pro veškerý obsah programu. Avšak pokud by se doprogramoval import textur za běhu programu a import modelů pro listy, bylo by snadné přidat tyto informace do vyexportovaného souboru.

Díky tomu, že Unreal Engine 4 se snaží být multiplatformní a různé platformy mají jiné funkce, chybí standardní dialog pro ukládání souborů. Proto jsem export nastavil do složky Output v místě instalace programu.

5.11.10 Parametry, jejich popis a doporučené hodnoty

Zkratka	Jméno parametru	Popis	Optimální hodnoty
A_X	Limity natočení listů v ose X	2D vektor, kde komponenta X určuje minimální odchylku od osy a komponenta Y maximální.	{-15,15}
A_Y	Limity natočení listů v ose Y	2D vektor, kde komponenta X určuje minimální odchylku od osy a komponenta Y maximální.	{-15,15}
A_Z	Limity natočení listů v ose Z	2D vektor, kde komponenta X určuje minimální odchylku od osy a komponenta Y maximální.	{-15,15}
S	Měřítko listu	2D vektor, kde komponenta X určuje měřítko a komponenta Y maximální.	{1,2}
M	Přídavek kolizního tělesa	Zvětší kolizní těleso o svou hodnotu a tím zabezpečí, že listy nebudou vrůstat do kolizního tělesa.	{velikost listu}
—	LeafMeshes	Pole obsahující statické modely listů.	—
—	Material	Materiál použitý na kmen stromu.	—

5.11.11 Poznámky

UE4 není standardně přizpůsoben mashům generovaným za chodu. Při některých hodnotách dochází ke špatnému výpočtu velikosti stínové mapy a způsobuje to pád aplikace. Tato chyba je závislá na hardware počítače a projevuje se pouze na některých sestavách. Záleží především na výrobci grafické karty a verzi ovladače k ní. Tato chyba je velmi těžce odstranitelná, protože k ní dochází ve zdrojovém kódu enginu. Návod na zprovoznění projektu je umístěn v příloze bakalářské práce na přiloženém DVD spolu se zdrojovými kódy a ukázkovým programem. Poslení verze zdrojových kódů programu je na GIT úložišti https://github.com/AmateurCz/TreeGenerator.

5.12 Použité nástroje

- Micosoft Visual Studio 2013
- Micosoft Visual Studio 2012
- Maxon CINEMA 4D R16
- Unreal Engine 4.7

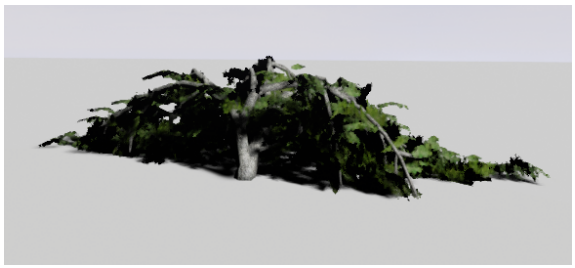
5.13 Změny vzhledu v závislosti na parametrech

Nejdůležitějším parametrem je počet iterací. Se zvyšováním jeho hodnoty dochází k více větvení a větvě jsou delší. Hodnoty nad 30 produkují velmi mnoho pupenů a výpočet může trvat i několik hodin.

Obrázek 20: Výsledek při počtu iterací 8

Obrázek 21: Výsledek při počtu iterací 18

Pomocí parametru gravitace $\vec{g}$ se ovlivňuje směr stáčení větví. Při záporných hodnotách parametrů v ose Z jsou větve přitahovány k zemi. Při vysoké záporné hodnotě se ztrácí vzhled stromu a rostlina se plazí po zemi. Při vysoké kladné hodnotě jsou větve prudce nahoru a strom je vysoký a úzký.

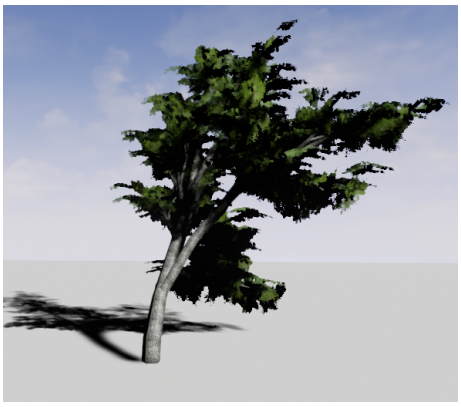


Obrázek 22: Výsledek při velké záporné gravitaci {0,0,-15}

Obrázek 23: Výsledek při velké kladné gravitaci {0,0,15}

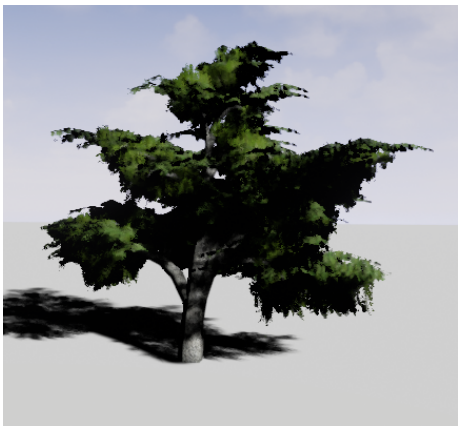
Síla světla L určuje hustotu stromu a délku segmentů. Délka segmentů je také ovlivňována parametrem síla prostředí E, který by měl být nepřímo úměrný parametru síla světla.

Obrázek 24: Výsledek při L=25, E=5



Obrázek 25: Výsledek při L=25, E=1

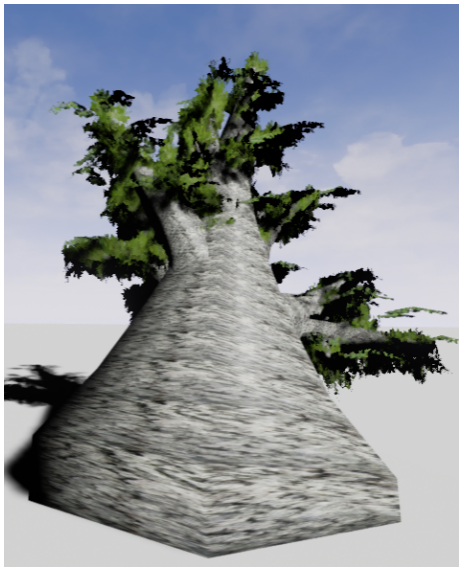
Obrázek 26: Výsledek při L=2, E=1



Obrázek 27: Výsledek při L=2, E=5

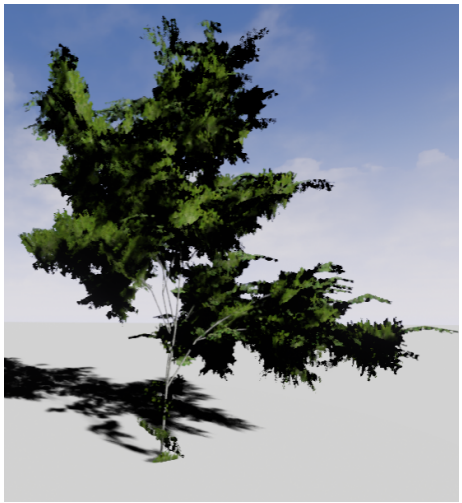
Parametrem tloušťka větve se ovlivňuje tloušťka koncových větví D. Parametrem C se ovlivňuje tloušťka následujících větví směrem ke kmeni.

Obrázek 28: Výsledek při D=3, C=3.5



Obrázek 29: Výsledek při D=3, C=1.5

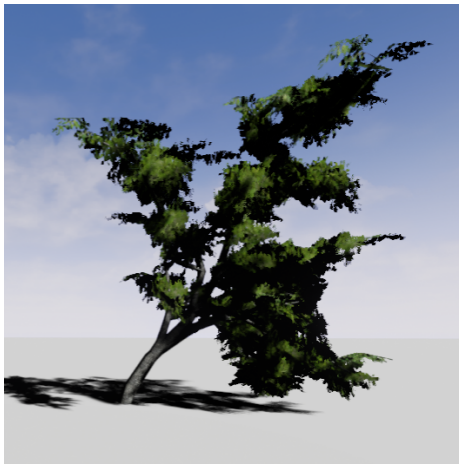
Obrázek 30: Výsledek při D=0.2, C=1.5



Obrázek 31: Výsledek při D=0.2, C=3.5

Parametr O je vzdálenost od pupene, ve které jsou zrušeny atraktory. Pokud nechceme, aby se křížily větve, je nutno dodržet podmínku, aby hodnota parametru O byla větší než násobek parametrů E a L. Tento parametr úzce souvisí s parametrem E_R, který ovlivňuje velikost oblasti, ve které atraktory ovlivňují směr růstů. Čím je větší rozdíl těchto dvou parametrů, tím dochází k menším odchylkám směru růstu větví.

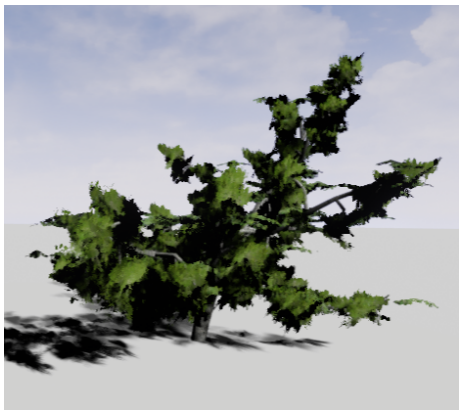
Obrázek 32: Výsledek při 0=200 a E_R=160



Obrázek 33: Výsledek při 0=80 a E_R=70

Parametr E_α ovlivňuje šířku kuželu, ve kterém atraktory ovlivňují směr růstu. Tím je zaručen maximální úhel, pod kterým vyrůstají sekundární větve.

Obrázek 34: Malý úhel vlivu E_α=25



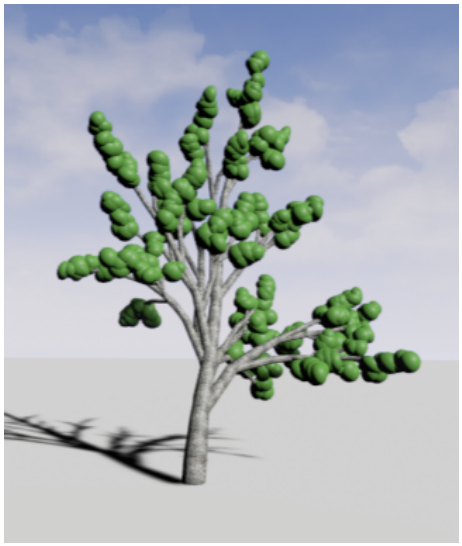
Obrázek 35: Velký úhel vlivu E_α=180

Parametr LeafMeshes určuje geometrii listu. Jedná se o pole, a díky tomu je možné přiřadit na jeden strom těchto geometrií několik. K výběru s dané množiny listů pak dochází náhodně. Geometrie určuje velikost, tvar, barvu, texturu listu.

Obrázek 36: Strom bez listů

Obrázek 37: Strom používající listy z knihovny programu SpeedTree

Obrázek 38: Strom s jednoduchými listy



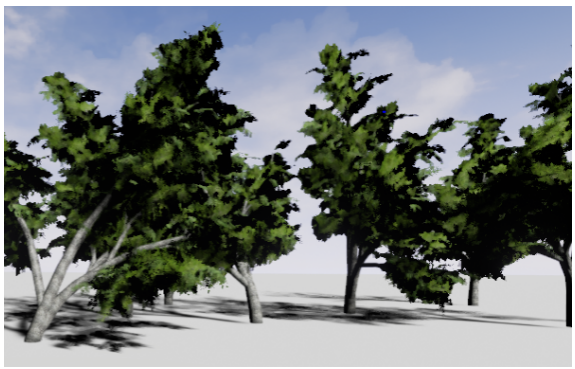
Obrázek 39: Strom s koulemi místo listů

Růst listů také ovlivňuje parametr D_{MAX} (5.9), který udává maximální tloušťku větve, na které rostou listy. Zabraňuje vygenerování listů na kmeni a hrubých větvích.

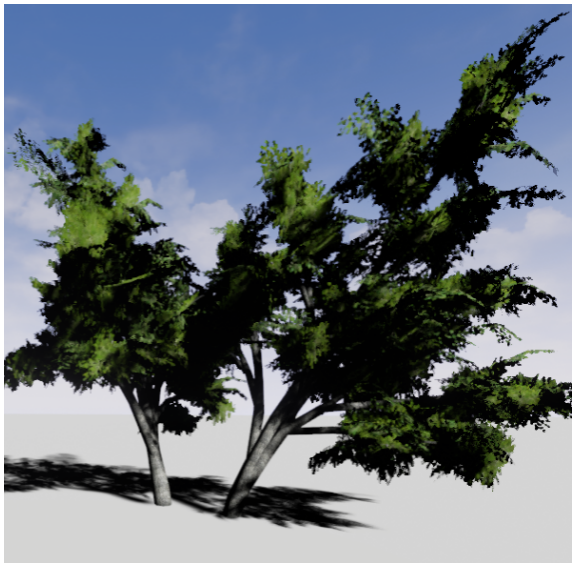
Obrázek 40: Řídký strom

Obrázek 41: Hustý strom

Pomocí parametru kořeny stromů definujeme polohu a počet stromů, které budeme generovat. Při generování více než čtyřiceti stromů dochází k velkému navýšení výpočetního času.



Obrázek 42: Les



Obrázek 43: Dva stromy blízko sebe

6 Závěr

Bakalářská práce splňuje své zadání ve všech jeho parametrech. Jejím úkolem bylo nastudovat metody generování rostlin a stromů založené na biologicky inspirovaných algoritmech. To je předmětem teoretické části práce. Vytvoření nástroje pro generování rostlin je popsáno v praktické části práce. Zpracovaný algoritmus produkuje realisticky vypadající a estetické stromy. Zároveň umožňuje několik způsobů, jak tento výsledek ovlivnit a přizpůsobit potřebám uživatele. Díky úpravám a použití jazyka C++ se mi podařilo udržet hardwarové a časové nároky na přijatelné úrovni. Použití jazyka C++ také umožňuje snadné portování kódu pro jiné populární nástroje jako např. Autodesk Maya, 3D Studio. Přitom je program otevřený dalším změnám a vylepšením.

Jako další vývoj této aplikace by bylo vhodné zakomponovat generování LOD a rozšířit animaci stromů do úrovně náporů větru. Lze přidat možnost importu objektů za běhu programu, aby nebyl nutný Unreal Editor. Druhou možností by bylo naimplementovat vlastní renderovací engine, a tím se zbavit závislosti na Unreal Engine.

7 Reference

[1] Arbaro.
URL http://arbaro.sourceforge.net

[2] Autodesk FBX SDK.
URL http://usa.autodesk.com/adsk/servlet/pc/item?siteID=123112&id=10775847

[3] Doupě.cz.
URL http://doupe.zive.cz/

[4] The Gnomon Workshop.
URL http://www.thegnomonworkshop.com/

[5] IFS.
URL http://web.comhem.se/solgrop/3dtree.htm

[6] Oficiální stránky MAXON.
URL http://www.maxon.net

[7] Plant generator list.
URL http://vterrain.org/Plants/plantsw.html

[8] Project C.A.R.S.
URL http://www.wmdportal.com/projectnews/

[9] runions.net.
URL http://adam.runions.net

[10] Speed Tree Blog.
URL http://blog.speedtree.com/

[11] What is Unreal Engine.
URL https://www.unrealengine.com/what-is-unreal-engine-4

[12] Wikimedia.
URL www.wikimedia.org

[13] XFrog.
URL http://xfrog.com/

[14] Adam Runions, P. P., Brendan Lane: Modeling Trees with a Space Colonization Algorithm. 9 2007.
URL http://algorithmicbotany.org/papers/colonization.egwnp2007.large.pdf

[15] Epic Games: *Dokumentace Unreal Engine 4*.
URL https://docs.unrealengine.com/latest/INT/

[16] MAXON: *Dokumentace CINEMA 4D API*.
URL https://developers.maxon.net

[17] Ping Tan, J. W. S. B. K. L. Q., Gang Zeng: Image-based Tree Modeling. 8 2007.
URL http://research.microsoft.com/en-us/um/people/sbkang/publications/TreeModeling07.pdf?q=imagebased-plant-modeling

[18] Wojciech Pałubicki, S. L. A. R. B. L. R. M. P. P., Kipp Horel: Self-organizing tree models for image synthesis. 2009.
URL http://algorithmicbotany.org/papers/selforg.sig2009.small.pdf